Article

International Journal of
HIGH PERFORMANCE
COMPUTING APPLICATIONS

The International Journal of High Performance Computing Applications
1–10
© The Author(s) 2013
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1094342013488258
hpc.sagepub.com
SAGE

Failure prediction for HPC systems and applications: Current situation and open issues

Ana Gainaru[1,2], Franck Cappello[2,3], Marc Snir[2,4] and William Kramer[1]

Abstract
As large-scale systems evolve towards post-petascale computing, it is crucial to focus on providing fault-tolerance strategies that aim to minimize fault's effects on applications. By far the most popular technique is the checkpoint–restart strategy. A complement to this classical approach is failure avoidance, by which the occurrence of a fault is predicted and proactive measures are taken. This requires a reliable prediction system to anticipate failures and their locations. One way of offering prediction is by the analysis of system logs generated during production by large-scale systems. Current research in this field presents a number of limitations that make them unusable for running on real production high-performance computing (HPC) systems. Based on our observations that different failures have different distributions and behaviours, we propose a novel hybrid approach that combines signal analysis with data mining in order to overcome current limitations. We show that by analysing each event according to its specific behaviour, our prediction provides a precision of over 90% and its able to discover about 50% of all failures in a system, result which allows its integration in proactive fault tolerance protocols.

Keywords
failure prediction, fault tolerance, signal analysis

1 Introduction

At the scale of today's large scale systems, fault tolerance is no longer an option, but a necessity. With a system mean time between failures (MTBF) of less than 1 day (Kindratenko, 2011) and predictions that future systems will experience delays of couple of hours between failures, current fault-tolerance strategies face serious limitations (Snir et al., 2011). A complement to current approaches is represented by failure avoidance, a technique which is based on an accurate failure predictor. Failure avoidance uses the information received by a failure predictor to facilitate proactive fault-tolerance mechanisms such as proactive job migration or proactive checkpoint.

There are two types of predictions that are possible for high-performance computing (HPC) systems. The first is state prediction where algorithms estimate the state of each node in the system. Current research uses diagrams in order to keep track of the states and the transitions between them in real time (Stearley et al., 2012; Chen et al., 2004; Salfner and Malek, 2007) and in general they use the states to decide whether a job can be scheduled on a specified node. The second type is represented by failure prediction where algorithms focus on providing information about when and where failures will occur in the near future. This paper focuses on the second type.

In general, failure prediction is based on the observation that there is a fault–errors–failure propagation graph (Salfner et al., 2010). The fault generates a number of errors that could be observable at the system level, which represent either notifications in the log files or changes in performance metrics. The propagation chain ends with the failure which is observed at the application level and usually is represented by an application interruption. However, the error could propagate and generate other effects such as performance degradation.

Our model is based on the observation that different failures have different distributions and create different

[1] National Centre for Supercomputing Applications, Urbana, IL, USA
[2] University of Illinois at Urbana-Champaign, Urbana, IL, USA
[3] INRIA, Rocquencourt, Le Chesnay Cedex, France
[4] Argonne National Laboratory, Argonne, IL, USA

Corresponding author:
Ana Gainaru, University of Illinois at Urbana-Champaign, Office 4017 NCSA, National Center for Supercomputing Aplications, 1205 W. Clark Street, Urbana, IL 61801, USA.
Email: againaru@illinois.edu

symptoms in the system. Current state-of-the-art research in the field of failure prediction for HPC systems is based on data-mining approaches and do not differentiate between the behaviour of distinctive failures. In our work, we introduce the concept of signal analysis in the context of event analysis, which allows us to characterize the behaviour of different events and to study how failures affect each of them. This paper highlights the limitations of current fault predictors and proposes ways of overcoming them by combining our signal analysis approach with existing data mining techniques. We show that by analysing each event according to its specific behaviour, our prediction provides a precision of over 90% and its able to discover about 50% of all failures in a system. The paper focuses on a detailed analysis of the prediction method by investigating the characteristics and bottlenecks of each step of the prediction process.

2 Related work

Over the years, approaches on failure prediction have been developed in relation to reliability theory and preventive maintenance (Gertsbakh, 2000; Nassar and Andrews, 1985; Schroeder and Gibson, 2006). Models evolved by trying to incorporate several factors into the distribution, for example the manufacturing process (Vilalta et al., 2002) or code complexity (Farr, 1996). However, all of these methods are tailored to long-term predictions and do not work appropriately for online failure prediction.

More recent methods for short-term failure prediction are typically based on runtime monitoring as they take into account a current state of the system. There are two levels of online failure prediction in literature: component-level and system-level failure prediction. The first level assumes methods that observe components (hard drive, mother board, DRAM, etc) with their specific parameters and domain knowledge and define different approaches that give best prediction results for each (Hwang et al., 2012). One example of this type of approach is to compare the execution of good components with failed ones. A couple of studies from different fields that fit in this category are Bolander et al. (2009); Patra et al. (2010). For the HPC community, one example is Zheng et al. (2007) in which matrices are used to record system performance metrics at every interval. The algorithm afterwards detects outliers by identifying the nodes that are far away from the majority.

The second level is represented by system-level failure prediction, in which monitoring daemons observe different system parameters (system log, scheduler logs, performance metrics, etc.) and investigate the existence of correlations between different events. In the last couple of years, a significant number of papers have been proposed that focus on providing predictions by analysing different HPC systems. However, most predictors are able to use the information extracted in the training phase for only short prediction span after which a new training phase is required. For example, Zheng et al. (2010) is using almost 3 months of training for predicting only half of month of execution. Another example of a current fault predictor is given by Yu et al. (2011) where the authors compare two failure prediction approaches and study the influence that the observation window has on the results. Gu et al. (2010) use a meta-learning predictor to chose between a rule-based method and a statistical method depending on which one gives better predictions for a corresponding state of the system. Another approach for analysing the logs is given by Nakka et al. (2011), who investigated both usage and failure logs.

The study presented by Zheng and Yu (2011) makes a difference between system and application failures. The authors use RAS logs and job logs for filtering out the failures that do not have any effect on jobs running in the system. This allows them to make a couple of interesting observations that could help future failure predictors. A more general approach is made by Rajachandrasekar et al. (2012) where the authors propose a middleware solution between the application and different analysis modules. Failure predictors and other decision-making engines that rely on distributed failure information can benefit from their framework to facilitate proactive fault-tolerance mechanisms such as preemptive job migration.

A different approach is given by Lou et al. (2010) and Xu (2009), where the authors investigate parameter correspondence between different application log messages for extracting dependencies among system components. Another approach using time-series analysis is presented by Wang et al. (2010) where the authors implement different processing methods, such as spike detection and subspace method in order to find outlier patterns which indicate anomalies in monitored systems.

There are a number of ways of building prediction modules for large-scale systems. In this paper we focus on analyzing only log files for the purpose of prediction. Log files give useful information about many components in the system, however, due to their large size and unstructured format they are very complex and cannot be analyzed manually. In general, current state-of-the-art research is using data mining algorithms for automating this process (Yu et al., 2011; Zheng et al., 2010; Gu et al., 2010; Nakka et al., 2011; Liang, 2006). Most of these algorithms are using the same workflow: they group the messages in the log file into categories, filter redundant events both in time and space, extract correlations between events based on the small filtered set of log messages and in the end use the correlations to predict future events or failures.

Each step from the workflow introduces imprecision or noise that influences the accuracy of the prediction. In this paper we are analyzing and characterizing this noise. We also propose a novel methodology that decreases the noise and is able to offer predictions that can be used on real production systems.

3 Methodology

Our methodology follows the workflow presented in Figure 1. The modules are divided into two major phases:

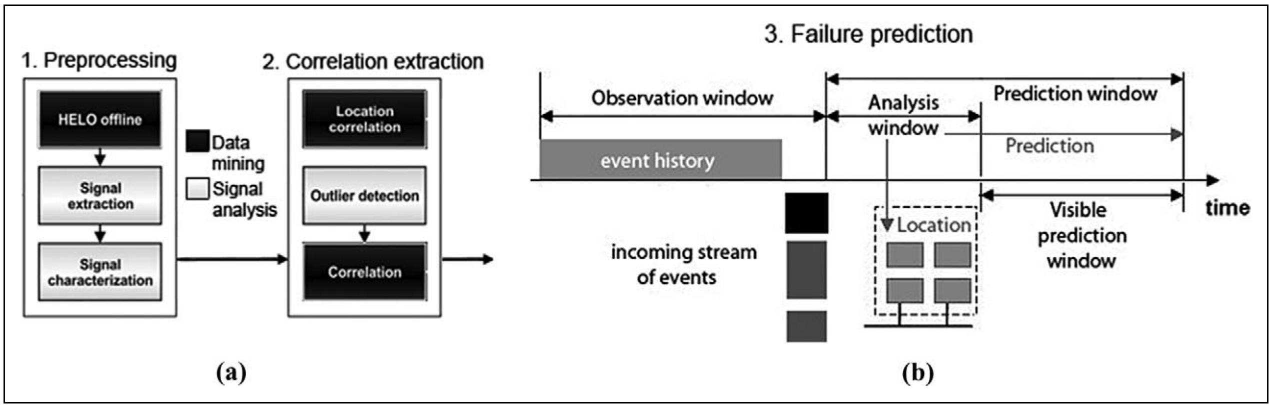

Figure 1. Failure prediction methodology: (a) offline training; (b) online prediction.

Table 1. Computing platform configuration and template examples.

System	Size (MB/day)	Rates (lines/minute)	Template example
BlueGene/L	5.74	15	node card is not fully functional
Spirit	55.58	339	log_error::is_request bad attempt to connect from * address n+
Mercury	152.4	868	node_bailout, d+ poll failed from node d+ * job will be killed
Current system	~ 2 GB/day	~ 5000	vm: killing process %s n+

offline characterization and online prediction. The first phase is called training phase and it works on historic event logs by first extracting all event types generated by the system and then by transforming the logs into time series interpreted as signals. Signal analysis allows us to characterize the behaviour of events affecting the system, highlighting the differences between failures. Data mining is an efficient method for extracting patterns in high-dimensionality sets so we use one of these methods to provide accurate correlations between defined behaviours to the online modules. The second phase is responsible with monitoring the incoming stream of events and deciding when to trigger a prediction. Also, modules in this phase are updating the correlations and the characteristics of event's behaviour to reflect the state of the entire system at different moments. We used the advantages of both data mining and signal analysis by offering a hybrid approach for predicting failures in HPC systems. We also implemented a propagation module that extends the prediction with location information so that the results could be applied for proactive migration or proactive uncoordinated checkpointing. The offline phase is described in more detail by Gainaru et al. (2012a) and the online by Gainaru et al. (2012b).

The following subsections follow the workflow of log analysis and prediction methods presented in the related work section. We will highlight the limitations of data mining algorithms and how our model overcomes them. At the end of each section we will discuss the weaknesses of our approach and future directions for optimizing them. Before starting the analysis, we define in this paragraph the main parameters that will be used in the following sections. Precision is seen as a measure of fidelity and represents the proportion of correctly found failures to all identified failures. Recall is the ratio of corrected identifications to all of the existing failures in the log and represents a measure of completeness. The lead time represents the time between when a prediction is triggered and when the failure occurs. The lead time can be used by fault-avoidance techniques to take a proactive action before the failure manifests in the system.

3.1 Group events of the same type in clusters

In the preprocessing step, we use the Hierarchical Event Log Organizer (HELO) (Gainaru et al., 2011) on the raw logs, resulting in a list of message templates that represent frequently occurring messages with similar syntactic patterns. Examples of logs and their characteristics can be found in Table 1. These templates represent regular expressions that describe different events in a system. In the online phase, we use HELO online to keep the set of templates updated and consistent to the output of the system.

Monitoring each event type separately is important since information regarding the events of interest might be hidden when the analysis is made at a lower granularity. For example, when looking at all types of failures at once, the logs show close to no spatial propagation. However, when analyzing only a certain type of filesystem errors, about 20% of failures affect only one node, the rest propagating on a variable number of locations (Heien et al., 2011).

For each of the event types, we use ELSA (Gainaru et al., 2012a) to map the number of occurrences per time step into the corresponding signal. After analyzing the signals for different HPC systems, we discovered that events are characterized by three types of signals presented in Figure 2(a): periodic, noise and silent. Usually, periodic signals represent

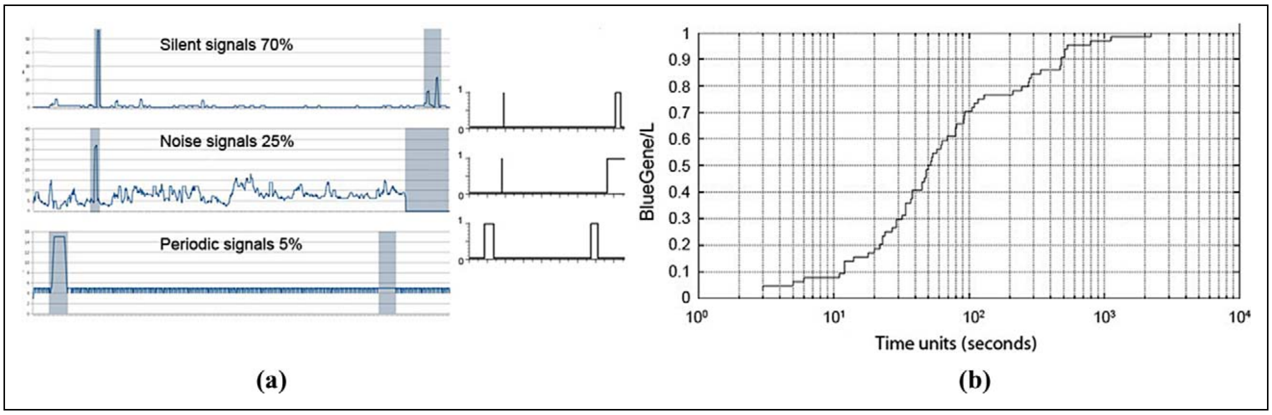

Figure 2. Signal analysis: (a) correlation methodology for each type of signal; (b) distribution of lead time.

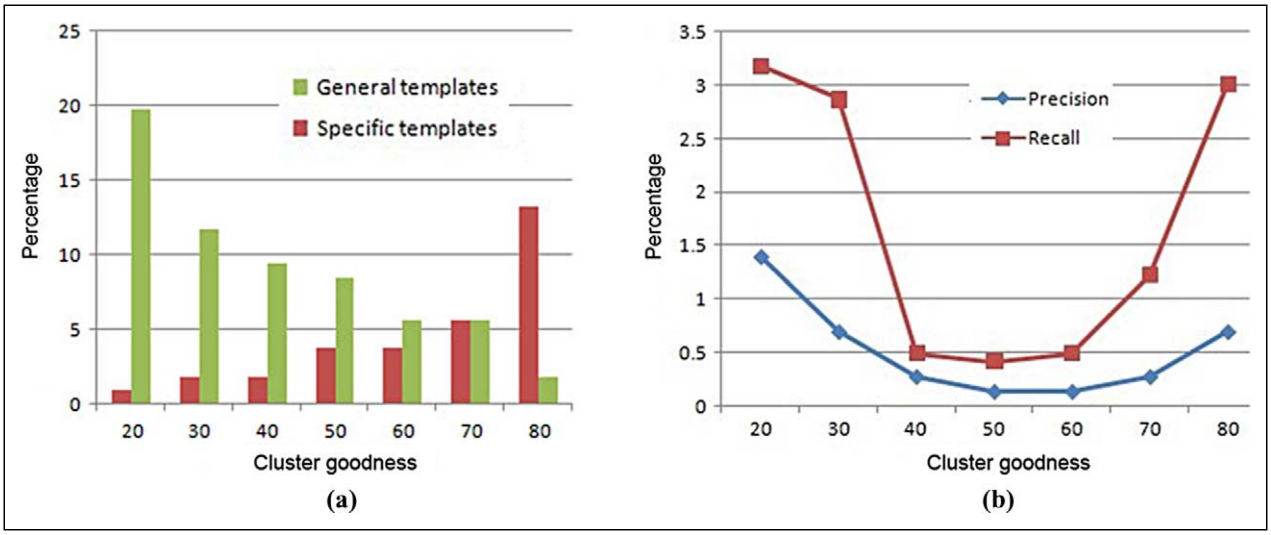

Figure 3. Preprocessing analysis: (a) percentage of incorrect templates; (b) precision/recall decrease.

events generated by monitoring daemons. Silent signals are defined as having a flat line around amplitude zero, and only from time to time presenting burst of messages and are usually characteristic for error messages, for example in case of PBS errors. However, sometimes normal messages behave the same and human knowledge is needed to make the difference between them. This is the case, for example, with "Job starting" messages. Noise are chatty signals that send notifications very often, both during the normal or faulty behaviour of the system. Anomalies in these event types usually represent a precursor to a failure in the system. One example are memory errors that cannot be corrected by ECC that present beforehand an abnormal number of correctable errors.

Discussion 1

Most data mining techniques rely on human expertise and cannot be used without this input in order to extract the event categories. Manually identifying categories is a time-consuming process and might result in category sets inconsistent across different systems. Moreover manual extraction usually generates wide granularity categories that affect the future analysis. Our focus is on providing an efficient way of identifying the event types that any system generates. In Gainaru et al. (2011) we showed that our tool can identify the correct templates with over 90% accuracy when compared with system administrators knowledge. Moreover, we analyze BlueGene/P that uses a different error code for each event type (e.g. e10000_ras_link_error for a specific type of link failure). We compared the templates generated by HELO with the error code list.

HELO generates templates that consist of constant words and variables. Variables identify manipulated objects or states for the program and are replaced by wildcards. In case constants are mistakenly replaced by wildcards the template becomes too general and, when variables are identified as constants by HELO, we call the corresponding templates too specific. We plotted the ratio of general and specific templates generated by HELO compared with the error codes of BlueGene/P in Figure 3(a) and how these differences affect the final prediction in Figure 3(b). Cluster goodness represents the similarity threshold that defines when two messages are part of one single events. Depending on the cluster goodness, there is a 2–30% difference between the

Table 2. System parameters.

MTBF	Failure distribution	Nodes	System lifespan	Propagation	Lead time
1 day	Weibull Scale=8116.7 Shape=0.387187	40,960	1 year	Yes 20% of failures	Weibull Mean = 50s

template set generated by HELO and the error codes of BlueGene/P. However, this is translated into a much smaller difference when looking at the impact on prediction, the highest impact showing a median difference of only 5% for recall and 3% for precision. Only for extreme values the impact is higher. We argue that this step affects in a small way the final prediction so automatic processes provide great benefits compared with human interaction without significantly affecting the final results.

Discussion 2

Because data mining is expensive by nature and the size of the log files are posing serious limitations on the analysis step, filtering is used to reduce the size of the analyzed dataset both in time and space without loosing the log's characteristics. One great advantage of using signal analysis is that it eliminates one of the steps in the workflow used by the pure data mining approaches. Filtering redundant events is no longer necessary since the signals compress the data in the log in a natural way without losing any occurrence. The limitation of filtering methods was analysed by DiMartino (2012) by creating a log generator that keeps the properties of real logs and that gives a ground truth with which to compare the results of different filtering techniques. Our observations show that the noise from this step filters out 7% to 12% of the events that might be useful to prediction.

3.2 Extract the correlation between events

In our experiments we observed that different failures show different distributions and behaviours. We use different signal analysis techniques to shape all of these behaviours and to characterize the way failures might affect them. This represents an important step since data mining algorithms apply the same extraction methods on all data entries.

The process for extracting correlations with ELSA is presented in Figure 1(a). In the first step, anomalies are extracted from each signal by first using wavelet functions to characterize the normal behaviour of each signal and then by monitoring changes in the signal's normal frequency and intensity. The right part of the figure shows the transformed signals after filtering out the normal behaviour. What is left for all signals are two values, zero when the messages are generated during the normal behaviour and one when an anomaly occurs. This process ensures that the data mining algorithms for extracting correlations are applied on the same type of data points. The algorithms in detail are described by Gainaru et al. (2012b)

To test the noise induced by this step we created a log generator that takes into account system parameters that were observed on Inrepid, the BlueGene/P system at ANL (Alam, 2008). Table 2 presents the parameters and their values that were used for the experiments. The log generator first creates failures in the log corresponding to the system parameters and then adds precursors based on a predefine correlation set. The correlations set represents the ground truth and is based on our past experiences with HPC systems. We used the values found after analysing Blue Gene/L from (Gainaru et al., 202b).

Discussion 3

The generated log is created so that all failures are predictable allowing the results obtained by using ELSA to represent the noise induced by the analysis process. Figure 4(a) shows the percentage of incomplete or completely missing sequences extracted by ELSA or by using a data mining technique, such as that in Zheng et al. (2010). The data mining algorithm that was used is the one incorporated in ELSA but applied on the raw log, by completely excluding all signal analysis modules from the process. We broke down the results by looking at correlations that incorporate different signal types. The figure shows that ELSA gives better results for correlations between noise signals or between signals that have different behaviours. Owing to this fact, overall ELSA cannot detect approximately 25% of the correlations that were used to construct the logs while the data mining approach has far worse results by losing about 52% of the correlations.

Discussion 4

In order to get a better understanding of the impact of log characteristics on prediction's results, we computed the precision and recall values by predicting both the synthetic log and also the logs from a real HPC system. Moreover, we analysed at the same time, how data mining algorithms behave compared with ELSA's results in Figure 4(b). Our previous observation is visible here as well, there is a difference of more than 20% of recall between ELSA and when only using data mining. Moreover, the recall value for ELSA is approximately 78% which can be explained by the fact that ELSA is not able to find almost a quarter of the initial correlations. This 22% represents the noise of the correlation extraction method used by ELSA and the noise given by the complexity of the logs. Interestingly only approximately a third of the correlations lost by ELSA are responsible for the high decrease

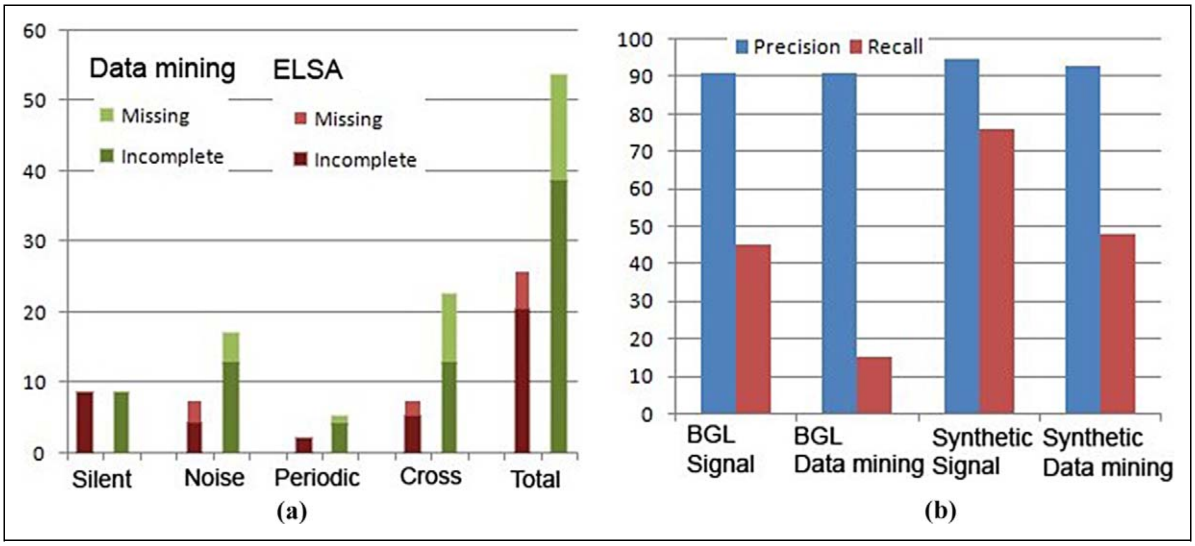

Figure 4. Correlation analysis: (a) percentage of incomplete/missing correlations; (b) precision/recall on BlueGene/L and synthetic logs.

in recall. In the future, we plan to isolate them and further study their properties.

Discussion 5

When looking at the results obtained on the logs generated by Blue Gene/L, we observe that the precision is close to that for synthetic logs, however the recall is only about 45% (Gainaru et al., 2012b). We consider that this 45% together with the rest of 22% up to the recall value obtained for synthetic logs represents the predictable set of failures from Blue Gene/L. The rest of 33% is represented by unpredictable failures represented by failures that do not offer precursors in the log file. This indicates performance metrics or other precursors detectors might give more information before a failure and could help in increasing the results of our predictor.

3.3 Failure prediction

Figure 1(b) shows the overview of the prediction process. The observation window is used to decide whether the current event is an anomaly. The analysis time represents the overhead of our method in making a prediction: the execution time for detecting the outlier, triggering a correlation sequence, and finding the corresponding locations. The time delay until the predicted event will occur in the system is defining the prediction window, which starts right after the observation point but is visible only at the end of the analysis time. The visible prediction window represents the lead time that a fault-avoidance technique might use. The values used for the lead time given by the correlations are presented in Figure 2(b). After the analysis using the log generator, we observed the lead time distribution shifts to the left which means we obtain shorter lead times. This is due to the correlations loss seen when using ELSA. The two problems are related and have the highest impact on prediction's result so we plan to analyse different correlation extraction methods into more detail in the future.

Discussion 6

All modules implemented in ELSA have online phases where they update the data generated in the training phase. In general, current research is not updating the correlations found offline and thus has limitations when using a short training set. We believe this limitation makes the prediction unrealistic when used on real production systems. To study the impact of not adapting the correlation set on the prediction's result, we used the data collected by ELSA during the training phase to predict the next 9 months of BlueGene/L and plotted the recall for each month in Figure 5(a). We have similar results when using the synthetic logs, however due to space limitation we did not present this figures. It is visible that the prediction keeps a high recall value only for the first couple of months and then decreases dramatically. By adapting the correlations and signal characterization over time we were able to keep the recall value almost constant throughout the entire studied life cycle of the system. The recall value presents a slight increases at a 3-month interval when ELSA is redoing the offline analysis. In the rest of the time, ELSA is relying only on the updates of existing correlations.

Discussion 7

For some fault-avoidance techniques the cost of predicting a failure that does not appear in the system is low compared with experiencing an unpredicted failure. This is, for example, the case of object migration with Charm++ (Zheng et al., 2004). Therefore, we did a study of the recall/precision trade-off in Figure 5(b). In general, the recall increases when using low threshold values for deciding when a

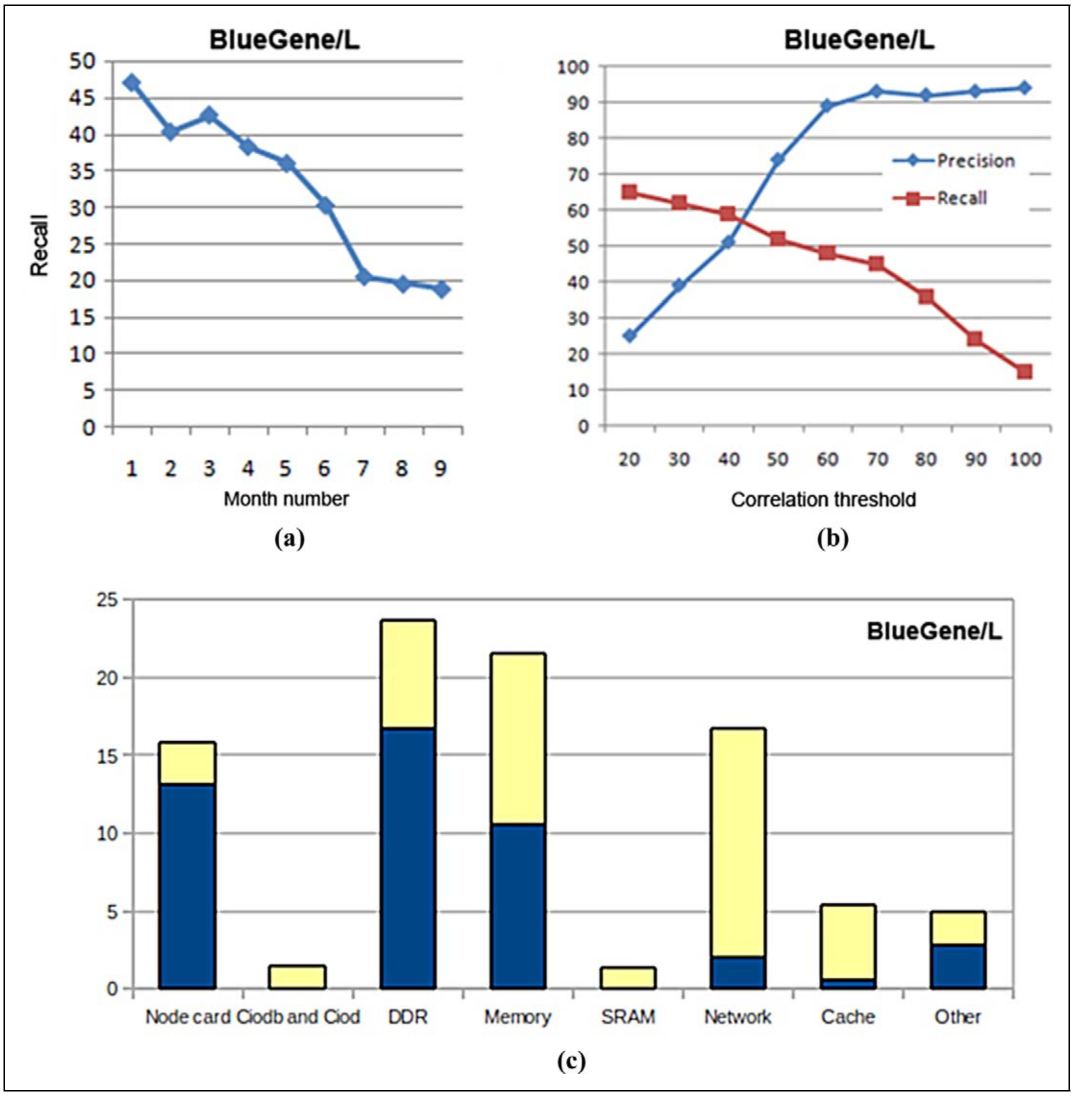

Figure 5. Prediction analysis: (a) recall on different months; (b) recall/precision tradeoff; (c) recall breakdown on components.

correlation is strong. Interestingly, the maximum recall value reached by ELSA is 63% which is close to what we observed in the previous section as being the amount of predictable failures. However, the cost in precision is really high, making more than 70% of ELSA's predictions wrong. It is also noticeable that the precision decreases at a higher rate than the increase in recall. Depending on the fault-avoidance technique different values might be the best option.

Discussion 8

In a more detailed analysis, we break down the predicted events into different categories. The results are presented in Figure 5(c), where each bar represents how often a certain type of error appears in the log as a percentage reported to all errors in the system. The dark portion of every bar represents the correctly predicted cases out of the total occurrences. We observed that the node card errors were the type that our system detected with a high rate; more than 80% of the occurrences were predicted. Overall, we observed an uneven distribution between different components in the system. For example the results obtained for network and cache failures are almost one order of magnitude lower than the results for network card. For the future, we plan to focus on analysing each component individually and try to understand what influences the prediction process for each of them.

4 Discussion

Accurate predictions are necessary for proactive fault-tolerance solutions. These solutions have the benefit of reducing the overhead due to fault-tolerance actions and the amount of lost work due to predicted failures. However, an extra overhead is added due to wrong predictions. The trade-off between this overhead and the benefit is highly influenced by the predictors recall and precision. We believe that understanding current prediction methods and their limitations is crucial in designing failure-avoidance techniques for exascale systems.

The correlation extraction method has the highest impact on prediction results. Therefore, the choice of the methodology is the utmost part of the prediction. We plan in the future to analyse various algorithms and study their results on large-scale production systems to get a better understanding of their limitations. Data mining algorithms have particularly poor results on noise and periodic signals. Our observation that failures do not affect the same way the system and are represented in different ways in system logs allowed us to analysed failures differently and, in the end, offer more accurate predictions.

Adapting the set of correlations and behaviour characterization is a necessity when working on real systems. Correlations using the last couple of months become unusable after less than 1 month of predictions. The pace of change is becoming increasingly fast for current and future HPC systems, so it is no longer viable for system administrators to give their input in any of the online analysis steps.

With the implementation of more accurate failures predictors there have been developed a number of mathematical models (Aupy et al., 2012; Gainaru et al., 2012b) that deal with characterizing the benefit of merging predictors with current checkpointing protocols. We combined ELSA with FTI (Bautista-Gomez et al., 2011), a fast multi-level checkpointing protocol and observed the overhead induced by the predictor is between 2% and 6%. When using the mathematical models with our predictor parameters and overhead, we observe that the benefit of this fault-avoidance technique can exceed 20% (Bouguerra et al., 2013). Another direction of our future work focuses on providing real implementations for different fault-avoidance protocols and computing their actual benefit when running large-scale applications in production.

5 Conclusions

Failure prediction has made outstanding progress in the last 5 years and for future HPC systems this technique could give benefits of over 20% when associated with different fault-avoidance techniques compared with the classical fault-tolerance approaches. Understanding the properties of a prediction module and exploiting them for enhancing fault-tolerance approaches and scheduling decisions is crucial for providing scalable solutions for dealing with failures on future HPC systems. In this paper, we described the problems and limitations faced in developing a accurate failure predictor. We show that a good solution is obtained by combining two different analysis techniques: signal analysis for shaping the normal behaviour of each event type and of the whole system and characterizing the way faults affect them and data mining for analysing the correlations between these behaviours. We analysed the deficiencies of our method and potential solutions in order to evolve our predictor into a viable solution for failure avoidance approaches. For the future, we plan to improve the results of our predictor by inspecting different precursor detectors to include in ELSA and by analysing in detail different error types for which our system has a low recall. Also, we will study to a wider extent, the practical way the prediction system influences current fault-tolerance mechanisms.

Funding

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. This research was done in the context of the INRIA-Illinois Joint Laboratory for Petascale Computing.

References

Alam S (2008) Early evaluation of IBM BlueGene/P. In: *Proceedings of 2008 International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 23.

Aupy G, Robert Y, Vivien F and Zaidouni D (2012) *Impact of Fault Prediction on Checkpointing Strategies*. Rapport de Recherche RR-8023, INRIA.

Bautista-Gomez L, Tsuboi S, Komatitsch D, Cappello F, Maruyama N and Matsuoka S (2011) FTI: high performance fault tolerance interface for hybrid systems. In: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*. New York: ACM Press, pp. 32:1–32:32. DOI: 10.1145/2063384.2063427.

Bolander N, Qiu H, Eklund N, Hindle E and Rosenfeld T (2009) Physics-based remaining useful life predictions for aircraft engine bearing prognosis. In: *Conference of the Prognostics and Health Management Society*.

Bouguerra MS, Gainaru A, Cappello F, Gomez LB, Maruyama N and Matsuoka S (2013) Improving the computing efficiency of hpc systems using a combination of proactive and preventive checkpointing. In: *Proceedings of IEEE IPDPS 2013*. IEEE Press.

Chen MY, Accardi A, Kýcýman E, et al. (2004) Path-based failure and evolution management. In: *Symposium on Networked Systems Design and Implementation*, vol. 1, p. 23.

DiMartino C (2012) Assessing time coalescence techniques for the analysis of supercomputer logs. In: *International Conference on Dependable System and Networks*, pp. 1–12.

Farr W (1996) Software reliability modeling survey. *Handbook of Software Reliability Engineering*. Highstown, NJ: McGraw-Hill, pp. 71–117.

Gainaru A, Cappello F and Kramer W (2012 a) Taming of the shrew: modeling the normal and faulty behavior of large-scale hpc systems. In: *Proceedings of IEEE IPDPS 2012*. IEEE Press.

Gainaru A, Cappello F, Snir M and Kramer W (2012b) Fault prediction under the microscope: a closer look into HPC systems. In: *Proceedings of 2012 International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press.

Gainaru A, Cappello F, Trausan-Matu S and Kramer B (2011) Event log mining tool for large scale HPC systems. In: *Proceedings of the 17th International Conference on Parallel Processing - Part I (Euro-Par'11)*. Berlin: Springer-Verlag, pp. 52–64.

Gertsbakh I (2000) *Reliability Theory: With Applications to Preventive Maintenance*. Berlin: Springer-Verlag.

Gu J, Zheng Z, Lan Z, White J, Hocks E and Park B-H (2010) Dynamic meta-learning for failure prediction in large-scale systems: A case study. *Journal of Parallel and Distributed Computing* 6: 630–643.

Heien E, Kondo D, Gainaru A, LaPine D, Kramer B and Cappello F (2011) Modeling and tolerating heterogeneous failures in large parallel systems. In: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. New York: ACM Press, p. 45.

Hwang AA, Stefanovici IA and Schroeder B (2012) Cosmic rays don't strike twice: understanding the nature of dram errors and the implications for system design. *SIGARCH Computer Architecture News* 40(1): 111–122. DOI: 10.1145/2189750.2150989.

Kindratenko V (2011) Trends in high-performance computing. *Computing in Science and Engineering* 3: 92–95.

Liang Y (2006) BlueGene/L failure analysis and prediction models. *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 425–434.

Lou J-G, Fu Q, Wang Y and Li J (2010) Mining dependency in distributed systems through unstructured logs analysis. *ACM SIGOPS Operating Systems Review* 44(1): 91–96.

Nakka N, Agrawal A and Choudhary A (2011) Predicting node failure in high performance computing systems from failure and usage logs. In: *IEEE Workshop on Dependable Parallel, Distributed and Network-Centric Systems*. IEEE Press.

Nassar FA and Andrews DM (1985) A methodology for analysis of failure prediction data. In: *IEEE Real-Time Systems Symposium*, pp. 160–166.

Patra AP, Bidhar S and Kumar U (2010) Failure prediction of rail considering rolling contact fatigue. *International Journal of Reliability, Quality and Safety Engineering* 17: 167.

Rajachandrasekar R, Besseron X and Panda DK (2012) Monitoring and predicting hardware failures in HPC clusters with FTB-IPMI. In: *International Parallel and Distributed Processing Symposium Workshops*.

Salfner F, Lenk M and Malek M (2010) A survey of online failure prediction methods. *Computing Surveys* 42: 1–42. DOI: 10.1145/1670679.1670680.

Salfner F and Malek M (2007) Using hidden semi-Markov models for effective online failure prediction. In: *Symposium on Reliable Distributed Systems*, pp. 161–174.

Schroeder B and Gibson G (2006) A large-scale study of failures in high-performance computing systems. *IEEE Transactions on Dependable and Secure Computing* 7(4): 337–351.

Snir M, Gropp W and Kogge P (2011) Exascale research: preparing for the post Moore era. *Computer Science Whitepapers*. Available at: https://www.ideals.illinois.edu/bitstream/handle/2142/25469/Exascale%20Research.pdf?sequence=2

Stearley J, Ballance R and Bauman L (2012) A state-machine approach to disambiguating supercomputer event logs. In: *Workshop on Managing System Automatically and Dynamically*, vol. 2, pp. 155–192.

Vilalta R, Apte CV, Hellerstein JL, Ma S and Weiss SM (2002) Predictive algorithms in the management of computer systems. *IBM Systems Journal* 41: 461–474.

Wang C, Talwar V, Schwan K and Ranganathan P (2010) Online detection of utility cloud anomalies using metric distributions. *Network Operations and Management Symposium*, pp. 96–103.

Xu W (2009) Online system problem detection by mining patterns of console logs. In: *IEEE International Conference on Data Mining*, pp. 588–597.

Yu L, Zheng Z, Lan Z and Coghlan S (2011) Practical online failure prediction for BlueGene/P: period-based vs event-driven. In: *IEEE Conference on Dependable Systems and Networks Workshops*, pp. 259–264.

Zheng GB, Shi L and Kale LV (2004) FTC-Charm++: An in-memory checkpoint-based fault tolerant runtime for Charm++ and MPI. In: *International Conference on Cluster Computing CLUSTER*. New York: ACM Press, pp. 93–103.

Zheng Z, Lan Z, Gupta R, Coghlan S and Beckman P (2010) A practical failure prediction with location and lead time for Blue Gene/P. In: *IEEE Conference on Dependable Systems and Networks Workshops*, pp. 15–22.

Zheng Z, Li Y and Lan Z (2007) Anomaly localization in large-scale clusters. In: *IEEE International Conference on Cluster Computing*, pp. 322–330.

Zheng Z and Yu L (2011) Co-analysis of RAS log and job log on Blue Gene/P. In: *Proceedings of the 2011 IEEE International Parallel and Distributed Processing Symposium*, pp. 840–851.

Author biographies

Ana Gainaru is a PhD student in the Computer Science Department of the University of Illinois at Urbana-Champaign. She is holding a research assistantship at the National Centre for Supercomputer Applications where she is part of the reliability and fault-tolerance team for the Blue Waters project. For the past 3 years her work in the context of the INRIA–UIUC Joint Laboratory for Petascale Computing has focused on fault analysis and prediction for large-scale systems. The toolkits developed, HELO and ELSA, have been applied on multiple HPC systems, are currently integrated into the Blue Waters system and are part of the first fault-tolerance hybrid solution that combines a checkpointing protocol with fault prediction. She holds a BS and MS in computer science from the Politehnica University of Bucharest where her work focused on analyzing how different data mining algorithms can be efficiently mapped on GPUs.

William T.C. Kramer is responsible for leading the Blue Waters project, a National Science Foundation-funded project at NCSA. Blue Waters is the most powerful general purpose computational and data analytics open science system available. Blue Waters supports a wide and diverse range of science and engineering investigations at unprecedented scale. Previously Kramer was the general manager of the NERSC at Lawrence Berkeley National Laboratory (LBNL) was responsible for all aspects of operations and customer service for NASA's Numerical Aerodynamic Simulator (NAS) supercomputer center. Blue Waters is the 20th supercomputer Kramer deployed and/or manages, deployed and managed large clusters of workstations, five extremely large data repositories, some of the World's most intense networks. He has also been involved with the design, creation and commissioning of six "best of class" HPC facilities. He holds a BS and MS in computer science from Purdue University, an ME in electrical engineering from the University of Delaware, a PhD in computer science at UC Berkeley. His research interests include large-scale system performance evaluation, systems management, resource management scheduling, fault detection and resiliency, and cyber protection. He has awards from NASA, Berkeley Laboratory, the Association for Computing Machinery (ACM) and was named one of HPCWire's "People to Watch" in 2005 and 2013 and Inside HPC first "Rockstar of HPC". He is the founder of several organizations, including the ACM/IEEE George Michael Memorial HPC Fellowship, the Open Science Grid Executive Committee and the DECUS Seminar Program.

Marc Snir is Director of the Mathematics and Computer Science Division at the Argonne National Laboratory and Michael Faiman and Saburo Muroga Professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. He currently pursues research in parallel computing. He was head of the Computer Science Department from 2001 to 2007. Until 2001 he was a senior manager at the IBM T.J. Watson Research Center where he led the Scalable Parallel Systems research group that was responsible for major contributions to the IBM SP scalable parallel system and to the IBM Blue Gene system. He received a Ph.D. in Mathematics from the Hebrew University of Jerusalem in 1979, worked at NYU on the NYU Ultracomputer project in 1980–1982, and was at the Hebrew University of Jerusalem in 1982–1986, before joining IBM. He was a major contributor to the design of the Message Passing Interface. He has published numerous papers and given many presentations on computational complexity, parallel algorithms, parallel architectures, interconnection networks, parallel languages and libraries and parallel programming environments. He is an Argonne Distinguished Fellow, AAAS Fellow, ACM Fellow and IEEE Fellow.

Franck Cappello is Program Manager and Senior Computer Scientist at Argonne National Laboratory. Before moving to ANL, he held a joint position at Inria and University of Illinois at Urbana Champaign where he initiated and co-directed from 2009 the INRIA–Illinois Joint Laboratory on Petascale Computing. Until 2008, he led a team at INRIA where he initiated the XtremWeb (Desktop Grid) and MPICH-V (fault-tolerant MPI) projects. From 2003 to 2008, he initiated and directed the Grid5000 project, a nationwide computer science platform for research in large-scale distributed systems. He has authored papers in the domains of fault tolerance, high-performance computing, desktop Grids and Grids and contributed to more than 70 program committees. He is editorial board member of the international *Journal on Grid Computing*, *Journal of Grid and Utility Computing* and *Journal of Cluster Computing*. He is a steering committee member of IEEE/ACM CCGRID. He was the Program program co-chair for ACM HPDC2014, Test of time award chair for IEEE/ACM SC13, Tutorial co-Chair of IEEE/ACM SC12, Technical papers co-chair at IEEE/ACM SC11, Program chair of HiPC2011, program cochair of IEEE CCGRID 2009, Program Area co-chair IEEE/ACM SC'09, General Chair of IEEE HPDC 2006.